Test QUDA with AMD GPUs on ROCm Platform

YUJIANG BI[1]

IHEP

April 5, 2019

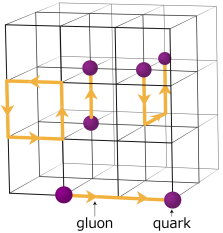
[1] byujiang@ihep.ac.cn, IHEP

Outline

Lattice QCD and QUDA

Quantum Chromodynamics (QCD)

- A theory describing strong interaction between quarks and gluons
- Asymptoic freedom, quark confinement and chiral symmetry spontaneous breaking
- Running coupling constant α_s to ∞ as energy to 0
- Perturbative method fails and non-perturbative required!!!



Lattice QCD

- A Non-perturbative QCD calculation framework.
- Continuous space-time to discrete space-time
- Monte Carlo simulating calculation
- Huge complex matrices: $\mathbf{4} \times \mathbf{3} \times \mathbf{3} \times \boldsymbol{T} \times \boldsymbol{L}^{\mathbf{3}} \approx \mathbf{9}\boldsymbol{G}$, T=L=64
- Massive computation resource required $\rightarrow$ CPU + GPU.

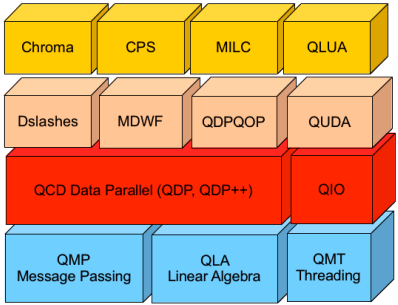
Lattice QCD and QUDA

What is QUDA?

- A widely used GPU framework for LQCD.
- Heavily based on CUDA and supported by NVIDIA.
- Easy to be integrated into existing programs.

And CHROMA?

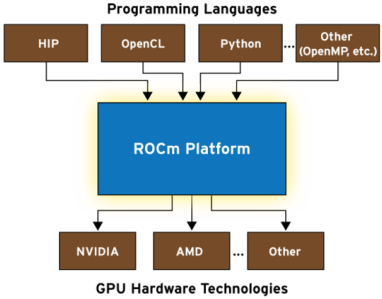
- A Common framework for LQCD simulations.
- Combinated with QUDA in order to using GPU.



ROCm and HIP TODO

ROCm - Radeon Open Compute Platform

- Fully Open Source GPU accelerator computing solution
- Hyper Scale: supports AMD & NVIDIA GPU..
- LLVM compiler as backend: strict grammar
- Programming Model: HIP, HCC and OpenCL.
- Born at 2016: BUGS but under fast development.
- Nearly SAME API and data structure as CUDA.



HIP, HCC, OpenCL

HIP

- Single-source Host+Kernel
- convert CUDA to portable C++
- C++ Kernel Language and C Runtime
- Supports AMD and NVIDA GPU
- Similar to CUDA
- Easy to port existing CUDA Codes.

HIP Key Features

- Supports for most commonly used CUDA APIs.
- Full C++ support including templates, namepace, lambdas...
- Portable to AMD/ROCm and NVIDIA/CUDA.
- hipify - automating tools for the codes translating.

HCC

- Single-source Host+Kernel
- C++ Kernel Language and C Runtime
- Pure C++ language.
- Supports only AMD GPU

OpenCL

- Split Host and Kernel
- C99 Kernel and C Runtime
- Supports CPU, GPU and FPGA
- Porting Existing OpenCL Codes.

Our Goal?

1. Porting QUDA from CUDA to ROCm platform.
2. Optimizing QUDA performance on ROCm platform.
3. Performing LQCD calculation using QUDA on AMD GPU.
4. Building QUDA on AMD or NVIDIA GPUs with ROCm.

Outline

Obstacles and Experiences

Obstacles

- QUDA is a BIG library of nearly 300,000 lines and heavily depends on CUDA.
- QUDA is under fast development, and being reconstructed in progress.
- ROCm and HIP are young and under fast development with bugs fixed every week.
- Depending on Eigen with incomplete support for ROCm/HIP.

Experiences

- A official portin guide - HIP Porting Guide.
- A official programming guide - HIP Programming Guide.
- Current supported API are listed in the official docs.
- Official tools - hipify-perl, hipify-cmakefile ... to automate the portin progress...
- Ported programs: Caffee, Tensorflow, PyTorch, Paddle, DNN, Thrust ...

CUDA Dependencies

QUDA depends on the following cuda libs and features:

CUDA	cublas	cub	thrus	cufft	curand	cusparse	texture	PTX
ROCm/HIP	hipblas	hipcub	hipthrust	rocfft	hiprand	hipsparse	texture	-

Porting Procedure

Procedure

1. hipify tools to automatically porting source and cmake files in place.
2. Finding rest cuda related terms and replacing with HIP terms.
3. Mostly replacing prefix **cuda**, **CU** or **cu** to **hip**.
4. Modifying cmake modules to using HIP and other libs.
5. Adding missing libs like **AMDDeviceLibs**.
6. Replacing CUDA related Macros like CUDA_ARCH with HIP_DEVICE_COMPILE

Examples

CUDA	CUmemorytype	CUresult	cuMemFree	cuCUdeviceptr	CUstream	...
ROCm/HIP	hipMemoryType	hipError_t	hipFree	hipDeviceptr_t	hipStream_t	...

Porting Problems

ROCm and Compiler

1. hipify does not cover all cuda terms, only a limited subset.
2. hipfiy could not handle function calls of multiple lines.
3. Can't implement some type conversion automatically, but CUDA can.
4. Order and account of functions argments are not consistent always.
5. Some API or TYPE not implemented yet: some cublas functions.
6. ROCm upgrading brings in unneccessary problems. **hipLaunchKernelGGL**
7. **cudaEventCreate** $\rightarrow$ **hipEventCreateWithFlags**

QUDA

1. Argments type in definition and implementation are not consistent.
2. Template functions omit default template type, not permitted in ROCm/HIP.
3. Under re-implementation using c++ template instead of cuda texture

Porting Problems

Example - const

```
// definition in quda_cuda_api.h
void qudaMemcpyAsync_(void *dst, const void *src, size_t count, hipMemcpyKind kind,
const hipStream_t &stream, const char *func, const char *file, const char *line);
// implementation in quda_cuda_api.cpp
void qudaMemcpyAsync_(void *dst, void *src, size_t count, cudaMemcpyKind kind,
const hipStream_t &stream, const char *func, const char *file, const char *line)
```

Example - teamplate type argment

```
// definition with full template type argments
 template <int mu, int nu, typename Float, typename Arg>
  __device__ __forceinline__ void computeFmunuCore(Arg &arg, int idx, int parity)
// calling with some argments omitted
case 0: computeFmunuCore<1,0,Float>(arg, x_cb, parity); break;
case 1: computeFmunuCore<2,0,Float>(arg, x_cb, parity); break;
```

Compiling and Linking

1. Many undetected cuda term not converted till compiling.
2. Up to 5 hours to fully compiling and linking quda once.
3. Undefined references due to inconsistent function definitions.
4. Division **double/Complex** errors when compiling..
5. Many undefined host or device functions in Eigen.
6. Some shared variables are Multi-defined or redifined.

Porting Results

- Modified Eigen to support QUDA.
- Static and shared quda lib: **libquda.a** & **libquda.so**.
- Excutable built-in testing quda programs.
- Compiled chroma with quda enabled.

Outline

QUDA Tests

- Could not run any tests of QUDA.
- Throwing out **No code available for function xxx** and exit.
- Or **Generating AMD GCN Kernel failed in llc for target: gfx906**.
- Could not optimize for loops..

Performance?

Sadly, no performance testing.

Still Under Way ....

Outline

Summary

Where We've Arrived?

1. Ported the latest QUDA to ROCm/HIP platform
2. Compiled QUDA with HIP and obtained libquda.{a,so}
3. Linked test programs with libquda
4. Compiled chroma with quda.

Dragons to Kill?

- Too long Compiling and linking speed...
- Uncomplete implementation of APIs of cublas, cuda driver and runtime...
- Compiler bugs like failing to generating device codes at runtime.
- GPU platform problems like too small texture size.

Outlook

1. Compiling runnable QUDA test and CHROMA.
2. Validating QUDA result with ROCm compared to that with CUDA.
3. Benchmarking QUDA module performance.
4. Perfroming real LQCD simulations and calculations. .
5. Optimizing QUDA on ROCm platform.
6. Reconstructing QUDA Using HIP to compiling on NVIDIA/AMD platforms.

Any Questions?

Special thanks to

QUDA

- M. A. Clark, R. Babich, K. Barros, R. Brower, and C. Rebbi, "Solving Lattice QCD systems of equations using mixed precision solvers on GPUs," Comput. Phys. Commun. 181, 1517 (2010) [arXiv:0911.3191 [hep-lat]].
- R. Babich, M. A. Clark, B. Joo, G. Shi, R. C. Brower, and S. Gottlieb, "Scaling lattice QCD beyond 100 GPUs," International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2011 [arXiv:1109.2935 [hep-lat]].
- M. A. Clark, A. Strelchenko, M. Cheng, A. Gambhir, and R. Brower, "Accelerating Lattice QCD Multigrid on GPUs Using Fine-Grained Parallelization," International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2016 [arXiv:1612.07873 [hep-lat]].
- M. A. Clark, A. Strelchenko, A. Vaquero, M. Wagner, and E. Weinberg, "Pushing Memory Bandwidth Limitations Through Efficient Implementations of Block-Krylov Space Solvers on GPUs," Comput. Phys. Commun. 233, 29 (2018) [arXiv:1710.09745 [hep-lat]].